

ĐÁP ÁN CHI TIẾT ĐỀ SỐ 7

Câu 1: ①

Câu 2: ②

Câu 3: ③

Câu 4: ②

Câu 5:

Cơ chế timeout \rightarrow dù R không thông báo cho A nhưng sau 1 khoảng thời gian A không nhận được ACK từ B vẫn truyền lại gói tin bị lỗi

Câu 6:

Sender gửi: $[0, 1, 2] \times 3 (\text{lần}) + [3, 4] \times 1 (\text{lần}) = 11 \text{ segment}$

Receiver xác nhận các gói theo thứ tự: $[0, 1, 2, 3, 4] = 5 \text{ ACK}$

Vậy cả A và B đã gửi: $11 + 5 = 16 \text{ segment}$

Câu 7: ①

Sender gửi: $[0] \times 3 (\text{lần}) + [1, 2, 3, 4] \times 1 (\text{lần}) = 7 \text{ Segment}$

Receiver xác nhận mỗi segment 1 ACK $0, 1, 2, 3, 4 = 5 \text{ ACK}$

\rightarrow Tổng số gói tin A và B đã gửi là $7 + 5 = 12$

Câu 8: ①

Câu 9:

Số segment cần gửi là 100 \rightarrow B cần xác nhận 50 ACK

Số segment bị lỗi là 10 \rightarrow B cần gửi $10 \times 2 \text{ ACK}$ để yêu cầu các segment này

\rightarrow Vậy tổng số ACK B phải gửi là $50 + 20 = 70 \text{ ACK}$

Câu 10: Congwin = 31

Câu 11:

Chuỗi nhị phân của các ký tự trong đoạn text "CAN"

là

$C(67) = 01000011$ $A(65) = 01000001$ $N(78) = 01001110$

Vậy UDP Checksum của đoạn text trên là

0100001101000001

0100111000000000

1001000101000001 \rightarrow Đảo bit có được UDP Checksum: 110111010111110

Câu 12: ②

Câu 13:

- Phân đoạn mạng 1 (MTU=1600) cần chuyển 5000 bytes data \rightarrow phải chia thành 4 datagram (d1, d2, d3 chuyển được $1580 \times 3 = 4740$ byte data; d4 chuyển nốt 260 bytes data cuối cùng).

- Ở phân đoạn mạng 2 (MTU=1300) mỗi datagram d1, d2, d3 bị chia thành 2 datagram nhỏ hơn là d11, d12, d21, d22, d31, d32 trong đó:

- + d11, d21, d31 mỗi datagram chuyển 1280 byte data.
 - + d12, d22, d32 mỗi datagram chuyển 300 byte data
 - + d4 khi đi qua phân đoạn này không bị phân mảnh (vì chỉ chứa 260 byte < 1300)
- Vậy:** B nhận tổng cộng 7 datagram, datagram thứ 5 chứa **1280** byte dữ liệu

Câu 14:

$A \rightarrow C \rightarrow D \rightarrow F$ (Giá trị: $1+5+2=8$)

Lưu ý: Sinh viên phải trình bày bảng tính toán các bước thực hiện giải thuật Dijkstra

Câu 15:

D _A	B	C
B	(2)	6
C	6	(2)
D	(4)	8
E	(3)	7
F	(6)	10

Câu 16: Ⓐ-ⓓ

Câu 17: Ⓑ

Câu 18: Mã của các ký tự trong chữ “HELP” là 72-69-76-80.

Chuỗi nhị phân tương ứng : 01001000 01000101 01001100 01010000

Ma trận kiểm tra chẵn lẻ 7x7

0	1	0	0	1	0	0
0	0	0	1	0	0	1
0	1	0	1	0	1	1
0	0	1	1	0	0	0
0	1	0	1	0	0	0
0	0	0	0	0	0	0
0	1	1	0	1	1	0

Câu 19: Mã ASCII của “p” là 112 → Mã nhị phân là: 01110000

→ mã Hamming của ký tự “n” là: 000111100000

Lưu ý: nếu SV áp dụng luật số lẻ thì mã hamming là 110011110000

Câu 20: Mã hamming lỗi 1 bit nhận được 00101010110

Xét vị trí 1: có 4 bit 1 tại các vị trí 1,3,5,7,9,11 → Không lỗi (bit parity =0)

- Xét vị trí 2: có 3 bit 1 tại các vị trí 2,3,6,7,10,11 → Lỗi (bit parity =1)

- Xét vị trí 4: có 2 bit 1 tại các vị trí 4,5,6,7 → Không lỗi (bit parity =0)

- Xét vị trí 8: có 2 bit 1 tại các vị trí 9, 10, 11 → Không lỗi (bit parity =0)

Chuỗi nhị phân vị trí bit bị lỗi là 0010 → vậy bit số 2 đảo lại thành 1

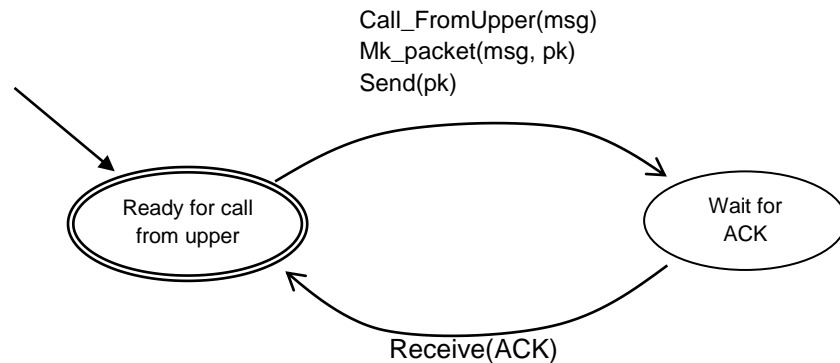
Chuỗi nhị phân sửa lại là: 01101010110

Mã nhị phân của ký tự bên gửi là: 1101110, ký tự gốc là n

Phần II – trả lời tự luận

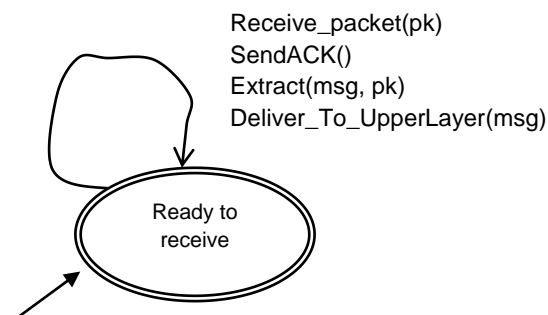
Tiếp cận theo giải pháp biên nhận tích lũy. Giải pháp biên nhận độc lập có thể được chấp nhận

Cơ bản dạng của FSM như sau. Mô tả cụ thể bên dưới



- Đường truyền từ A→B không tin cậy nên mỗi lần nhận yêu cầu từ tầng trên thì A sẽ phải gửi dữ liệu vào đường truyền, đồng thời thiết lập timer để đợi ACK của gói vừa gửi và chuyển sang trạng thái chờ xác nhận từ B
- Khi nhận được xác nhận ACK của gói đã gửi từ B thì trạng thái của A lại chuyển về sẵn sàng cho lời gọi kết tiếp từ tầng trên.
- Khi hết khoảng thời gian timer được thiết lập mà A không nhận được ACK của nó=> A gửi lại gói đã gửi trước đó
- Khi nhận được các ACK trùng lặp (3 ACK) A sẽ biết gói nào bị mất và gửi lại gói đã mất đó.

Cơ bản FSM của B có dạng như sau. Mô tả chi tiết ở dưới



B ở trạng thái đợi dữ liệu từ tầng dưới.

Nếu B nhận được gói dữ liệu đúng thứ tự => B đẩy gói dữ liệu vừa nhận và những gói được buffer trước đó lên tầng trên; tạo và gửi ACK biên nhận cho các gói dữ liệu đã nhận đúng
 Nếu B nhận được gói dữ liệu không đúng thứ tự (thường là cao hơn) => B sẽ buffer gói dữ liệu vừa nhận được, đồng thời tạo và gửi lại A một ACK biên nhận cho những gì mình đã nhận đúng (sẽ tạo ra ACK trùng lặp tại A).

Do đường truyền từ B→A là tin cậy nên mỗi lần dữ liệu tới B sẽ gửi xác nhận ACK về cho A (xác nhận này chắc chắn tới A).